Flexible Budgets



AdWords lets you set a daily budget amount for each campaign. However, some marketing initiatives will have a fixed cost associated with them, e.g., "I want to spend $5000 leading up to our fall sale". The bidding strategy gives you some control over how the daily budget is spent, but no control over how the budget is consumed during the campaign.

For example, if we want to spend only $5000 to advertise our fall sale and we want to advertise for 10 days, we could set a daily budget of $500 to use up the entire budget. However, this assumes that we will spend the entire amount each day AND we wish to spend it evenly. It's not possible to tell AdWords that you want to spend the bulk of your budget during the last few days.

This script will dynamically adjust your campaign budget daily with a custom budget distribution scheme.

How it works

**Testing budget strategies**

The script includes testing code to simulate the effects of running for multiple days. This gives you a better idea of what will happen when the script is scheduled to run daily over a period of time.

By default, this script will simulate an even budget distribution of $500 spent over 10 days.

function main() {
  testBudgetStrategy(calculateBudgetEvenly, 10, 500);
//  setNewBudget(calculateBudgetEvenly, CAMPAIGN\_NAME, TOTAL\_BUDGET, START\_DATE, END\_DATE);
}

The setNewBudget function call is commented out, which means we'll only run the testing code. Here is the output from the example:

Day 1.0 of 10.0, new budget 50.0, cost so far 0.0

Day 2.0 of 10.0, new budget 50.0, cost so far 50.0

Day 3.0 of 10.0, new budget 50.0, cost so far 100.0

Day 4.0 of 10.0, new budget 50.0, cost so far 150.0

Day 5.0 of 10.0, new budget 50.0, cost so far 200.0

Day 6.0 of 10.0, new budget 50.0, cost so far 250.0

Day 7.0 of 10.0, new budget 50.0, cost so far 300.0

Day 8.0 of 10.0, new budget 50.0, cost so far 350.0

Day 9.0 of 10.0, new budget 50.0, cost so far 400.0

Day 10.0 of 10.0, new budget 50.0, cost so far 450.0

Day 11.0 of 10.0, new budget 0.0, cost so far 500.0

Each day we calculate a new budget to make sure we're spending the budget evenly each day. After we've exceeded the budget allotted for the initiative, we set the budget to zero, halting spend.

You can change the budget strategy used by changing which function is used, or modifying the function itself. The script comes with two pre-built strategies:calculateBudgetEvenly and calculateBudgetWeighted; we've just tested the former—update the testBudgetStrategy line to use the other:

testBudgetStrategy(calculateBudgetWeighted, 10, 500);

Click **Preview** and check the logger output. Notice that this budget strategy allocates less budget early in the period and more during the last few days.

You can use this test method to simulate changes to the budget calculation functions and try your own approach to distributing a budget.

**Allocating a budget**

Let's look more closely at the calculateBudgetWeighted budget strategy:

function calculateBudgetWeighted(costSoFar, totalBudget, daysSoFar, totalDays) {
  var daysRemaining = totalDays - daysSoFar;
  var budgetRemaining = totalBudget - costSoFar;
  if (daysRemaining <= 0) {
    return budgetRemaining;
  } else {
    return budgetRemaining / (2 \* daysRemaining - 1) ;
  }
}

This function takes four arguments:

* costSoFar - how much has this campaign accrued in cost from the START\_DATE to today.
* totalBudget - how much we want to spend from START\_DATE to END\_DATE.
* daysSoFar - how many days have elapsed from START\_DATE to today.
* totalDays - the total number of days between START\_DATE and END\_DATE.

You can write your own function as long as it takes these arguments. Using these values, you can compare how much money you've spent so far against how much to spend overall and determine where you currently are within the timeline for the entire budget.

In particular, this budget strategy figures out how much budget remains (totalBudget - costSoFar) and divides that by twice the number of days remaining. This weights the budget distribution towards the end of the campaign. By using the cost since START\_DATE, it also takes into account "slow days" where you don't spend the entire budget you set.

**Budgeting for real**

Once you're happy with your budget strategy, you'll need to make a few changes before you can schedule this script to run daily.

First, update the constants at the top of the file:

* START\_DATE - set this to the start of your budget strategy—should be the current date or a day in the past.
* END\_DATE - set this to the last day you want to advertise with this budget.
* TOTAL\_BUDGET - the total amount you're trying to spend. This value is in account currency and may be exceeded depending on when the script is scheduled to run.
* CAMPAIGN\_NAME - the name of the campaign to apply the budget strategy to.

Next, disable the test and enable the logic to actually change the budget:

function main() {
//  testBudgetStrategy(calculateBudgetEvenly, 10, 500);
  setNewBudget(calculateBudgetWeighted, CAMPAIGN\_NAME, TOTAL\_BUDGET, START\_DATE, END\_DATE);
}

Scheduling

Schedule this script to run daily, at or shortly after midnight in the local timezone so as to direct as much as possible the upcoming day's budget. Note, however, that retrieved reports data such as cost could be delayed by about 3 hours, so the costSoFar parameter may be referencing yesterday's total for a script that is scheduled to run after midnight.

Setup

* Create a new AdWords script with the source code below.
* Save the script and click the Preview button. This script will (by default) simulate a budget strategy with $500 over 10 days. The logger output will reflect the day being simulated, the allocated budget for that day and the total amount spent to date.

[Creating an AdWords script](https://developers.google.com/adwords/scripts/docs/solutions/flexible-budgets)

Source code

var START\_DATE = new Date('April 1, 2013');
var END\_DATE = new Date('May 1, 2013');
var TOTAL\_BUDGET = 500;
var CAMPAIGN\_NAME = 'Special Promotion';

function main() {
  testBudgetStrategy(calculateBudgetEvenly, 10, 500);
//  setNewBudget(calculateBudgetEvenly, CAMPAIGN\_NAME, TOTAL\_BUDGET, START\_DATE, END\_DATE);
}

function setNewBudget(budgetFunction, campaignName, totalBudget, start, end) {
  var today = new Date();
  if (today < start) {
    Logger.log('Not ready to set budget yet');
    return;
  }
  var campaign = AdWordsApp.campaigns().
      withCondition('CampaignName = "' + campaignName + '"').
      get().
      next();
  var costSoFar = campaign.getStatsFor(dateToString(start), dateToString(end)).getCost();
  var daysSoFar = datediff(start, today);
  var totalDays = datediff(start, end);
  var newBudget = budgetFunction(costSoFar, totalBudget, daysSoFar, totalDays);
  campaign.setBudget(newBudget);
}

function calculateBudgetEvenly(costSoFar, totalBudget, daysSoFar, totalDays) {
  var daysRemaining = totalDays - daysSoFar;
  var budgetRemaining = totalBudget - costSoFar;
  if (daysRemaining <= 0) {
    return budgetRemaining;
  } else {
    return budgetRemaining / daysRemaining;
  }
}

function calculateBudgetWeighted(costSoFar, totalBudget, daysSoFar, totalDays) {
  var daysRemaining = totalDays - daysSoFar;
  var budgetRemaining = totalBudget - costSoFar;
  if (daysRemaining <= 0) {
    return budgetRemaining;
  } else {
    return budgetRemaining / (2 \* daysRemaining - 1) ;
  }
}

function testBudgetStrategy(budgetFunc, totalDays, totalBudget) {
  var daysSoFar = 0;
  var costSoFar = 0;
  while (daysSoFar <= totalDays + 2) {
    var newBudget = budgetFunc(costSoFar, totalBudget, daysSoFar, totalDays);
    Logger.log('Day %s of %s, new budget %s, cost so far %s', daysSoFar + 1, totalDays, newBudget, costSoFar);
    costSoFar += newBudget;
    daysSoFar += 1;
  }
}

/\*\*
 \* Returns number of days between two dates, rounded up to nearest whole day.
 \*/
function datediff(from, to) {
  var millisPerDay = 1000 \* 60 \* 60 \* 24;
  return Math.ceil((to - from) / millisPerDay);
}

function dateToString(date) {
  return date.getFullYear() + zeroPad(date.getMonth() + 1) + zeroPad(date.getDate());
}

function zeroPad(n) {
  if (n < 10) {
    return '0' + n;
  } else {
    return '' + n;
  }
}